Umeå University
Department of Computing Science
TDBC86 — Principles of Database Systems
Examination: December 21, 2006

Name (printed): ____

Personnummer (Swedish ID number): ____________________________

UNIX user ID: ____

Signature: ____

1. All answers must be written in English.
2. A pocket engineering calculator (miniräknare) and English/X - X/English dictionary may be used. No other help materials are allowed.
3. The examination consists of two parts. In Part I, which includes Problems 1–2, on pages 1–5, the answers **must** be written on the sheets provided. In general, only the answers will be checked for these questions; derivations need not be provided and will not be evaluated. In Part II, which includes Problems 3–10, the answers must be written on separate sheets.
4. You must turn in this cover sheet, the answer sheets for Part I (pages 1–5), and the pages containing your answers for Problems 3–10. Collate the sheets in numerical order of the problems, and write on only one side of the paper. Write your name and Swedish ID number on the top of each sheet. For questions 3–10, write the question number on the top of each sheet as well.
5. The examination has a total of 800 points, which constitute 80% of the course grade. The other 20% is based upon the obligatory exercises, as explained in the course syllabus. In order to pass the course, you must obtain at least 400 points on the examination, regardless of how many points you earn on the obligatory exercises.
6. For Problems 3–10, you have the choice, for each part, to do the problem, or to skip it for partial credit. In the table below, place an X in the position for any problem for which you have attempted a solution, and which you wish to have graded. It is extremely important that you fill in this table properly, because of the following option. For any box which is left blank, the associated question will not be graded, and you will instead be awarded 15% of the points for that question. Your decision to leave a box blank is definitive, so be very careful. For example, If you leave box 8(b) blank, your answer to that question will not be graded, even if it is completely correct. On the other hand, if you place an X in box 8(b), but provide no answer whatsoever to that question, you will not receive 15% of the points for that question. It is strongly recommended that you use a pencil, in case you change your mind! *Note that Problems 1–2 do not carry this option. No points are awarded for skipping them.*

Prob	3	4	5	6	7	8	9	10
(a)								
(b)								
(c)								
(d)								
(e)								
(f)								

Name and ID number: ______________________________

(1: 120 points total; 10 points for each part) Important: The answers to this question must be indicated below; do not write the answers to this question on a blank sheet.

The following relational schema with relation $R[ABCDEFG]$ and functional dependencies $\mathcal{F} = \{A \to DF, CD \to BG, E \to ABFG, G \to EF\}$ applies to all twelve parts of this problem.

Given below are eight projections of $R[ABCDEFG]$. Assume that each such projection is constrained by exactly the functional dependencies in $\mathcal{F}^+$ which embed in the projection. In the space provided, give all of the candidate keys for the projection, as well as a minimal cover (basis) of the functional dependencies from $\mathcal{F}^+$ which embed in it. Also, circle *all* of the normal-form properties which the projection possesses (even if some are implied by others). Circle the word "None" just in case none of the normal-form properties holds.

(a) Projection: $R[ABCDEFG]$

Candidate keys: ______________________________

Minimal FD cover (basis): ______________________________

Normal forms: None 2NF 3NF BCNF

(b) Projection: $R[ABDEFG]$

Candidate keys: ______________________________

Minimal FD cover (basis): ______________________________

Normal forms: None 2NF 3NF BCNF

(c) Projection: $R[ABCD]$

Candidate keys: ______________________________

Minimal FD cover (basis): ______________________________

Normal forms: None 2NF 3NF BCNF

(d) Projection: $R[ACDF]$

Candidate keys: ______________________________

Minimal FD cover (basis): ______________________________

Normal forms: None 2NF 3NF BCNF

(e) Projection: $R[AEF]$

Candidate keys: ______________________________

Minimal FD cover (basis): ______________________________

Normal forms: None 2NF 3NF BCNF

Name and ID number: ______________________________

(f) Projection: $R[BEG]$

Candidate keys: ______________________________

Minimal FD cover (basis): ______________________________

Normal forms: None 2NF 3NF BCNF

(g) Projection: $R[CDF]$

Candidate keys: ______________________________

Minimal FD cover (basis): ______________________________

Normal forms: None 2NF 3NF BCNF

(h) Projection: $R[CDG]$

Candidate keys: ______________________________

Minimal FD cover (basis): ______________________________

Normal forms: None 2NF 3NF BCNF

Shown below are four decompositions of the schema on the previous page. For each decomposition, indicate whether or not it has the properties of *lossless* and *dependency preserving* by circling the appropriate word(s). Circle the word *neither* just in case it is neither lossless nor dependency preserving.

In checking the lossless property, use only pairwise reconstructions (*i.e.*, join trees). Do not test for cyclic decompositions (*i.e.*, decompositions which are lossless, but the join can not be reconstructed pairwise) [It was not explained in the course how to test for such decompositions, so if you do not understand what this means, just use the techniques which were presented in the course.]

(i) Decomposition: $\{R[ABDEFG], R[CDF]\}$.
Neither Lossless Dependency preserving

(j) Decomposition: $\{R[BEG], R[AEF]\ R[CDG]\}$.
Neither Lossless Dependency preserving

(k) Decomposition: $\{R[BEG], R[AEF], R[CDG], R[ABCD]\}$.
Neither Lossless Dependency preserving

(l) Decomposition: $\{R[BEG], R[AEF], R[ACDF], R[CDG]\}$.
Neither Lossless Dependency preserving

(2: 80 points) A database contains four data objects, A, B, C, and D. Six transactions are given on these objects, specified as follows.

T_1	T_2	T_3	T_4	T_5	T_6
Read(A)	Write(A)	Read(C)	Read(C)	Read(B)	Read(B)
Write(A)	Read(C)	Write(C)	Write(C)	Read(C)	Write(B)
Read(B)	Write(C)	Read(B)	Read(D)	Write(B)	
Write(B)		Write(B)	Write(D)		
Read(C)					
Write(C)					

Shown to the right is a schedule for this set of transactions, including a marked point at which a crash occurs. Using this schedule, answer the questions given on the following two pages. The following points are relevant:

- In the simple model here, only the pure immediate-update protocol and the pure deferred-update protocol will be considered. There is no DBMS cache, and so no need for checkpoints.
- Assume that all active transactions abort at the time of a crash, but that the log is preserved without damage. Regarding the state of the database itself, there are two scenarios. In the first (parts (a) and (b)), the state of the database is restored to its value before these transactions began, and recovery begins from that point. In the second (parts (c) and (d)), the database state remains exactly as it was at the point of the crash, and recovery must proceed from that point.
- The choice "No further action is necessary" (parts (c) and (d) only) is appropriate when the transaction had started before the time of the crash, but no special action on it is necessary for crash recovery. (It may or may not have committed before the time of the crash.)
- A re-do of a transaction from log entries is always to be preferred to a re-run. Thus, when both will result in a correct recovery, the correct answer is "Re-do from log entries".

T_1:	Begin
T_1:	Read(A)
T_1:	Write(A)
T_4:	Begin
T_4:	Read(C)
T_2:	Begin
T_2:	Write(A)
T_4:	Write(C)
T_4:	Read(D)
T_4:	Write(D)
T_4:	Commit
T_2:	Read(C)
T_2:	Write(C)
T_2:	Commit
T_1:	Read(B)
T_1:	Write(B)
T_3:	Begin
T_3:	Read(C)
T_3:	Write(C)
T_1:	Read(C)
T_1:	Write(C)
T_1:	Commit
T_5:	Begin
T_5:	Read(B)
T_5:	Read(C)
T_5:	Write(B)
T_5:	Commit
T_6:	Begin
T_6:	Read(B)
T_6:	Write(B)
T_6:	Commit
	Crash
T_3:	Read(B)
T_3:	Write(B)
T_3:	Commit

In answering parts (a) and (b) below, use exactly schedule shown on page 3, assuming that at the point of the crash, all active processes abort and the database state is restored to the value which it had just before T_1 began.

(a) To recover from the system crash in the context of the *pure immediate-update protocol*, for each transaction, indicate which of the following actions is applicable by placing an ╳ in the appropriate ☐, as so ☒. For each transaction, only one choice is correct.

T_1: ☐ Re-do from log entries.
☐ Re-run; no rollback necessary.

T_2: ☐ Re-do from log entries.
☐ Re-run; no rollback necessary.

T_3: ☐ Re-do from log entries.
☐ Re-run; no rollback necessary.

T_4: ☐ Re-do from log entries.
☐ Re-run; no rollback necessary.

T_5: ☐ Re-do from log entries.
☐ Re-run; no rollback necessary.

T_6: ☐ Re-do from log entries.
☐ Re-run; no rollback necessary.

(b) To recover from the system crash in the context of the *pure deferred-update protocol*, for each transaction, indicate which of the following actions is applicable by placing an ╳ in the appropriate ☐, as so ☒. For each transaction, only one choice is correct.

T_1: ☐ Re-do from log entries.
☐ Re-run; no rollback necessary.

T_2: ☐ Re-do from log entries.
☐ Re-run; no rollback necessary.

T_3: ☐ Re-do from log entries.
☐ Re-run; no rollback necessary.

T_4: ☐ Re-do from log entries.
☐ Re-run; no rollback necessary.

T_5: ☐ Re-do from log entries.
☐ Re-run; no rollback necessary.

T_6: ☐ Re-do from log entries.
☐ Re-run; no rollback necessary.

In answering parts (c) and (d) below, use exactly schedule shown on page 3, assuming that at the point of the crash, all active processes abort but that the database state remains as it was at the time of the crash.

(c) To recover from the system crash in the context of the *pure immediate-update protocol*, for each transaction, indicate which of the following actions is applicable by placing an ╳ in the appropriate ☐, as so ☒. For each transaction, only one choice is correct.

T_1: ☐ No further action is necessary. ☐ Re-do from log entries.
☐ Re-run; no rollback necessary. ☐ Re-run; rollback necessary.

T_2: ☐ No further action is necessary. ☐ Re-do from log entries.
☐ Re-run; no rollback necessary. ☐ Re-run; rollback necessary.

T_3: ☐ No further action is necessary. ☐ Re-do from log entries.
☐ Re-run; no rollback necessary. ☐ Re-run; rollback necessary.

T_4: ☐ No further action is necessary. ☐ Re-do from log entries.
☐ Re-run; no rollback necessary. ☐ Re-run; rollback necessary.

T_5: ☐ No further action is necessary. ☐ Re-do from log entries.
☐ Re-run; no rollback necessary. ☐ Re-run; rollback necessary.

T_6: ☐ No further action is necessary. ☐ Re-do from log entries.
☐ Re-run; no rollback necessary. ☐ Re-run; rollback necessary.

(d) To recover from the system crash in the context of the *pure deferred-update protocol*, for each transaction, indicate which of the following actions is applicable by placing an ╳ in the appropriate ☐, as so ☒. For each transaction, only one choice is correct.

T_1: ☐ No further action is necessary. ☐ Re-do from log entries.
☐ Re-run; no rollback necessary. ☐ Re-run; rollback necessary.

T_2: ☐ No further action is necessary. ☐ Re-do from log entries.
☐ Re-run; no rollback necessary. ☐ Re-run; rollback necessary.

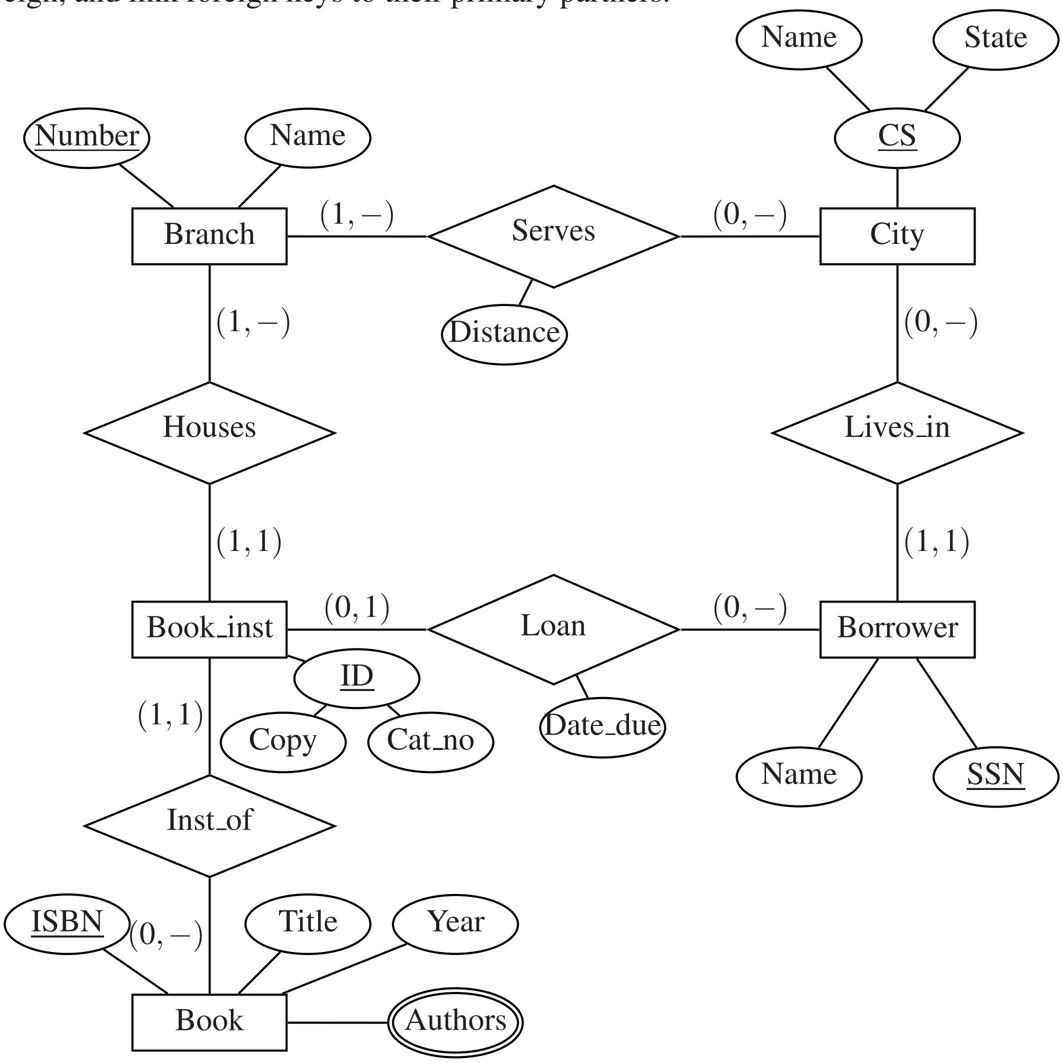
T_3: ☐ No further action is necessary. ☐ Re-do from log entries.
☐ Re-run; no rollback necessary. ☐ Re-run; rollback necessary.

T_4: ☐ No further action is necessary. ☐ Re-do from log entries.
☐ Re-run; no rollback necessary. ☐ Re-run; rollback necessary.

T_5: ☐ No further action is necessary. ☐ Re-do from log entries.
☐ Re-run; no rollback necessary. ☐ Re-run; rollback necessary.

T_6: ☐ No further action is necessary. ☐ Re-do from log entries.
☐ Re-run; no rollback necessary. ☐ Re-run; rollback necessary.

(3: 80 points) Shown below is an ER-diagram for a library database. Using the techniques developed in the course, map this diagram into an equivalent relational schema. Show all keys, primary and foreign, and link foreign keys to their primary partners.



- If a primary and/or foreign key consists of more than one attribute, make sure that your notation identifies and links these composite keys as groups.
- Use design decisions which produce the simplest relational schema. If there is a choice between creating an additional relation and adding attributes to an existing one, choose the latter. Similarly, a foreign key which cannot take on null values is to be preferred to one which can.

Notes:

- The (x, y) notation gives the minimum and maximum number of times that a given instance of the entity may participate in the relationship. Thus, $(1, 1)$ means exactly one, and $(0, -)$ means any number.
- Book_inst = book instance; Inst_of = instance of; cat_no = catalog number; CS = city state.
- A *book* is an entity with an ISBN; *e.g.*, the course textbook.
- A *book instance* is a physical book; *e.g.*, your copy of the course textbook.

(4: 80 points total; 20 points for each part) Answer the following questions concerning the *DBMS buffer* (also called the *DBMS cache*).

(a) In the context of writing the buffer to the permanent database, explain the difference between a *steal strategy* and a *no-steal strategy*.

(b) In the context of writing the buffer to the permanent database, explain the difference between a *force strategy* and a *no-force strategy*.

(c) Explain briefly how the process of *checkpointing* is complicated by the use of a steal strategy, as opposed to the use of a no-steal strategy.

(d) Explain briefly how the process of *checkpointing* is complicated by the use of a no-force strategy, as opposed to the use of a force strategy.

(5: 80 points total)

(a: 40 points) Give precise definitions of the following notions:

- (i) Two phase locking protocol (2PL).
- (ii) Strict two-phase locking protocol.
- (iii) Conservative two-phase locking protocol.
- (iv) Rigorous two-phase locking protocol.

(b: 20 points) Explain the importance of the two-phase locking (2PL) protocol.

(c: 10 points) Briefly explain the added advantages and disadvantages of a conservative 2PL strategy.

(d: 10 points) Identify a situation in which it is an advantage to enforce a strict 2PL strategy (over simple 2PL). Justify your answer with a brief but clear explanation.

(6: 80 points total) The following parameters apply to a relation which is to be stored in a B^+-tree. Assume that each node, interior or leaf, is stored in one page, and that each tuple is stored as one record.

Number of tuples in the relation	2×10^{10}
Size of one tuple	256 bytes
Size of pointer to a B^+-tree node	8 bytes
Size of internal B^+-tree key	24 bytes
Page size	4096 bytes
Sequential-access pointers in the leaf nodes	16 bytes total

Answer the following. To receive credit, you must show your work, including the formulas used. (If you show just numerical computations, without an underlying formula, full credit will not be awarded.)

(a: 10 points) Compute the maximum number of indices per non-leaf node.

(b: 10 points) Compute the maximum number of records per leaf node.

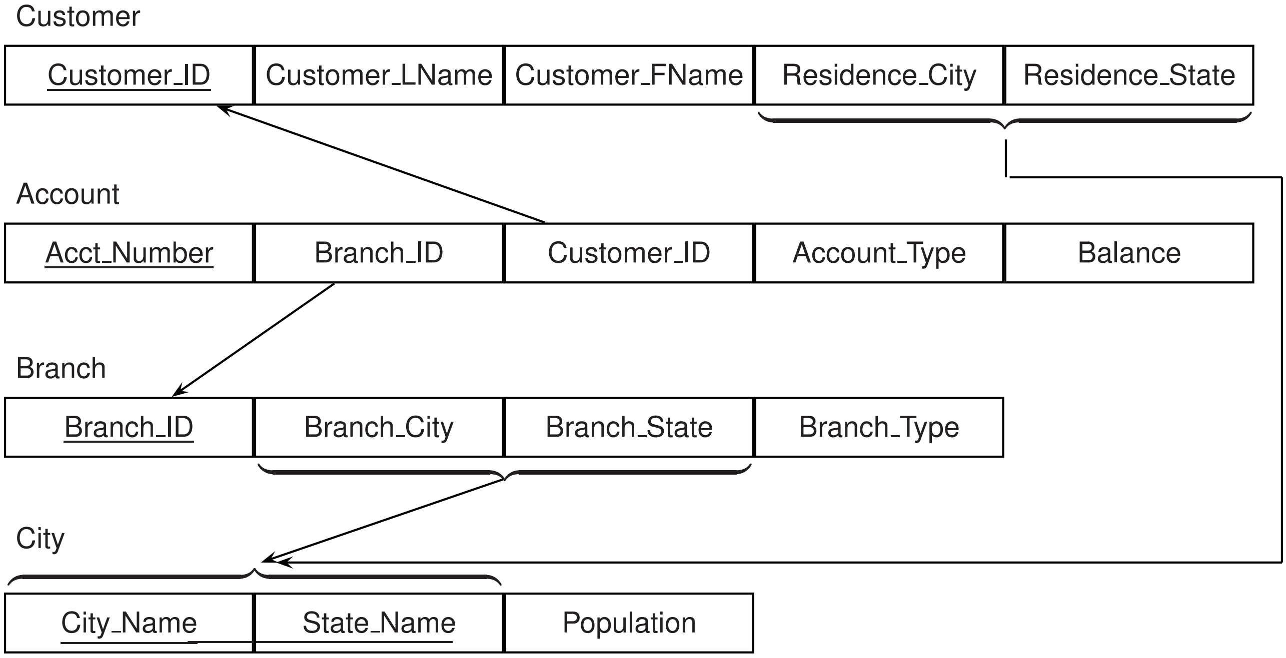
(c: 10 points) Compute the maximum depth that such a B^+-tree may have. (The depth is defined as the length of a path from the root to a leaf.)

(d: 10 points) Compute the minimum depth that such a B^+-tree may have.

(e: 20 points) For a B^+-tree of the maximum depth computed in (c), compute the average number of indices in the interior nodes other than the root, assuming that the root contains the minimum possible number of indices, and that the leaf nodes contain the minimum possible number of records. State whether or not this value is consistent with the constraints on the minimum and maximum number of indices in an interior node of a B^+-tree. In addition, if this value is consistent, compute the number of interior nodes and the number of leaf nodes, as well as the average number of records in each leaf node.

(f: 20 points) Repeat (e), this time assuming that the leaf nodes contain the maximum number of records possible. All other parameters remain the same as specified in (e).

The following schema applies to Problems 7 through 10. In each case, the primary key is underlined, and the arrows run from foreign keys to their primary counterparts.



The following queries apply to Problems 7 through 9, and reference the schema defined above.

(a) Find the names of those cities (together with the corresponding state) which house some branch, but which are not the residence of any customer.

(b) Find the names of those cities (together with the corresponding state) which house a branch of every type which is listed in the database. It may be assumed that the Branch relation is nonempty.

(c) Find the names (last and first) of those customers who have exactly two different types of accounts. (The user may have many accounts, but all of these accounts must be of one of exactly two types.)

Note: In the above schema, the attributes Customer_LName and Customer_FName identify the last and first names of the customer, respectively.

(7: 60 points total; 20 for each part) Solve each of the three queries (a) - (c) in the relational algebra. Functional operators, such as count and average, may not be used. Use the join notation described in the course notes: (The symbol $\bowtie$ is used to denote join; in the absence of subscripts it denotes the natural join; any other join conditions must be specified explicitly using subscripts.)

(8: 60 points total; 20 for each part) Solve each of the three queries (a) - (c) in the tuple relational calculus. Functional operators, such as count and average, may not be used.

(9: 60 points total; 20 for each part) Solve each of the three queries (a) - (c) using SQL. To keep the solutions simple and consistent, the following restrictions apply:

(i) Directives involving the keyword JOIN may not be used. Express the join conditions directly using conditions in the WHERE clause.

(ii) The solution must consist of a single SQL directive. Creation and subsequent use of temporary tables is not permitted.

(iii) Subqueries involving SELECT are allowed only in the WHERE and HAVING fields of queries and subqueries, and in the SET fields of UPDATE commands.

(iv) Aggregate operators, such as COUNT, SUM, AVG, MAX, and MIN may **not** be used.

(10: 100 points total; 20 points for each part) Provide solutions, in SQL, to the following additional queries. Restrictions (i)-(iii) from Problem 7 apply here as well. On the other hand, aggregate operations may be used, but only in the SELECT and HAVING clauses of queries and subqueries.

(a) Find the customer ID, as well the first and last names, of those customers who have a total balance of at least 1000 in all of their accounts.

(b) For each (City_Name,State_Name) pair in the City relation, list the total number of customers who live in the city defined by that (City_Name,State_Name) pair, the total number of accounts which are held by such customers, and the average balance over all accounts held by such customers. If there are no accounts associated with a given (City_Name,State_Name) pair, list the average as zero.

(c) Display the name of the state which has the greatest total balance over all accounts which are held in branches in that state; display the total balance as well. In case of a tie, list all such states.

(d) For each account which is housed in a branch which is located in the same state as the residence state of the customer who owns the account, add 10% to the total balance.

(e) Delete all branches which do not house any accounts.

Appendix

A useful formula for uniform (m,r,d) B-trees:

$$R(m,r,d) = (m+1) \times (r+1)^d - 1$$

with

$$\begin{aligned}
m &= \text{the total number of indices at the root node.} \\
r &= \text{the total number of records in each non-root node.} \\
d &= \text{the depth of the tree; } \textit{i.e.}, \\
& \text{the length of a path from the root to a(ny) leaf node.} \\
R(m,r,d) &= \text{the total number of records in the tree.}
\end{aligned}$$

A useful formula for uniform (m,q,r,d) B$^+$-trees:

$$R(m,q,r,d) = (m+1) \times (q+1)^{d-1} \times r$$

with

$$\begin{aligned}
m &= \text{the total number of indices at the root node.} \\
q &= \text{the total number of indices in each non-root, non-leaf node.} \\
r &= \text{the total number of records in each leaf node.} \\
d &= \text{the depth of the tree; } \textit{i.e.}, \\
& \text{the length of a path from the root to a(ny) leaf node.} \\
R(m,q,r,d) &= \text{the total number of records in the tree.}
\end{aligned}$$